ROSALES, Angel Abegail B.

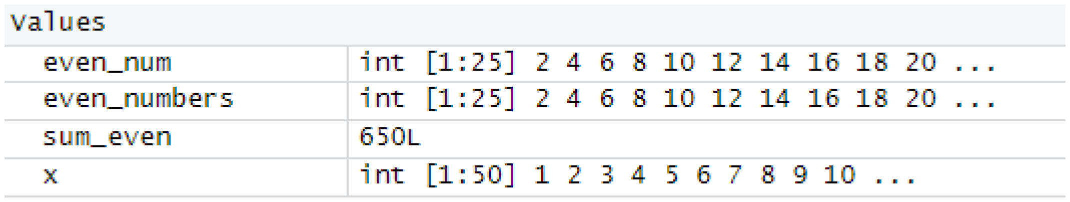
BSIT 3-5

Activity 3: Working with Vectors and Data Frames

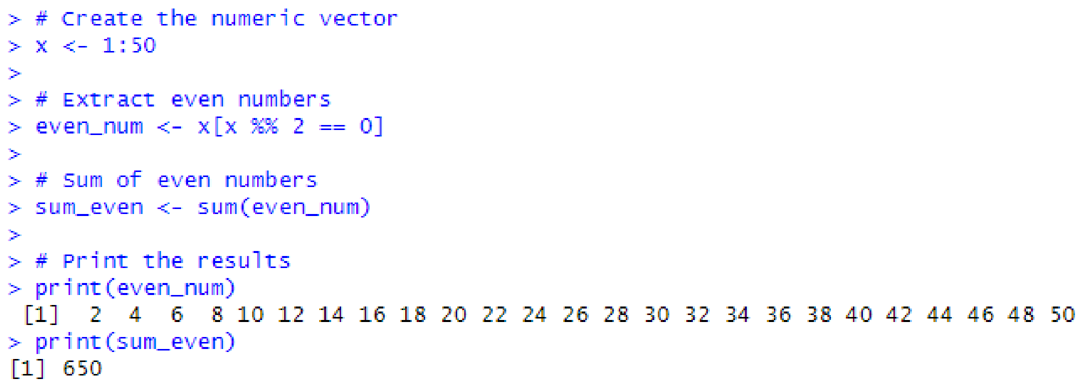
Instructions

Perform the tasks below using R programming. Write the corresponding R code for each question and explain the output briefly. Ensure your code is properly formatted and test it in RStudio or an equivalent IDE.

1. Create a numeric vector x with values from 1 to 50. Extract all even numbers and calculate their sum using a combination of indexing and functions.



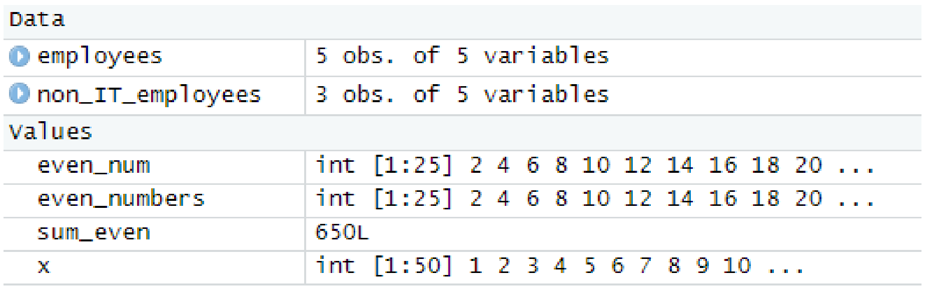
Values	
even_num	int [1:25] 2 4 6 8 10 12 14 16 18 20 ...
even_numbers	int [1:25] 2 4 6 8 10 12 14 16 18 20 ...
sum_even	650L
x	int [1:50] 1 2 3 4 5 6 7 8 9 10 ...



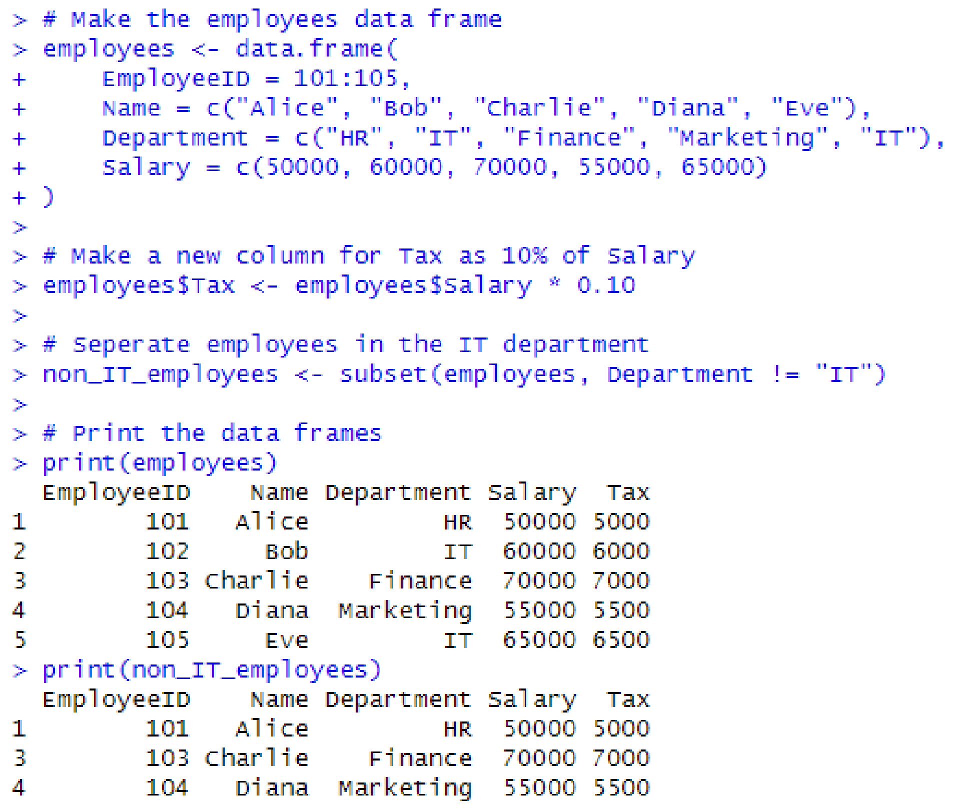
```
> # Create the numeric vector
> x <- 1:50
>
> # Extract even numbers
> even_num <- x[x %% 2 == 0]
>
> # Sum of even numbers
> sum_even <- sum(even_num)
>
> # Print the results
> print(even_num)
 [1]  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
> print(sum_even)
[1] 650
```

The expression x %% 2 == 0 checks for even numbers in x. Using x[x %% 2 == 0] retrieves those even numbers. The sum () function adds up these numbers. The output displays the even numbers and their total, which is 650.

2. Construct a data frame employees with columns:
   EmployeeID: 101 to 105
   Name: "Alice", "Bob", "Charlie", "Diana", "Eve"
   Department: "HR", "IT", "Finance", "Marketing", "IT"
   Salary: 50000, 60000, 70000, 55000, 65000
   Write code to:
   Calculate and add a new column Tax as 10% of Salary.
   Filter out employees in the IT department.



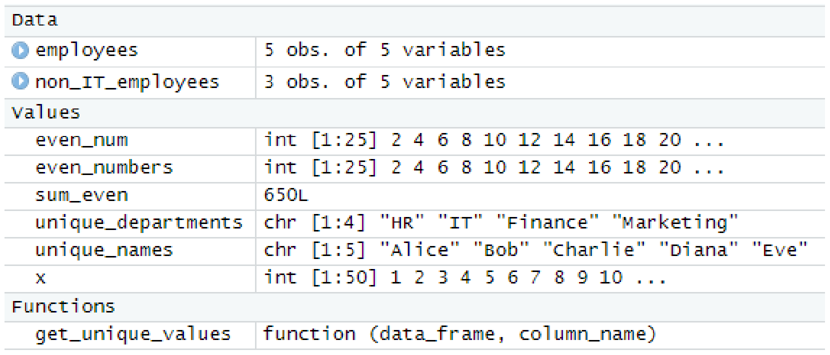
Data	
employees	5 obs. of 5 variables
non_IT_employees	3 obs. of 5 variables
Values	
even_num	int [1:25] 2 4 6 8 10 12 14 16 18 20 ...
even_numbers	int [1:25] 2 4 6 8 10 12 14 16 18 20 ...
sum_even	650L
x	int [1:50] 1 2 3 4 5 6 7 8 9 10 ...



```
> # Make the employees data frame
> employees <- data.frame(
+     EmployeeID = 101:105,
+     Name = c("Alice", "Bob", "Charlie", "Diana", "Eve"),
+     Department = c("HR", "IT", "Finance", "Marketing", "IT"),
+     Salary = c(50000, 60000, 70000, 55000, 65000)
+ )
>
> # Make a new column for Tax as 10% of Salary
> employees$Tax <- employees$Salary * 0.10
>
> # Seperate employees in the IT department
> non_IT_employees <- subset(employees, Department != "IT")
>
> # Print the data frames
> print(employees)
  EmployeeID    Name Department Salary  Tax
1        101   Alice         HR  50000 5000
2        102     Bob         IT  60000 6000
3        103 Charlie    Finance  70000 7000
4        104   Diana  Marketing  55000 5500
5        105     Eve         IT  65000 6500
> print(non_IT_employees)
  EmployeeID    Name Department Salary  Tax
1        101   Alice         HR  50000 5000
3        103 Charlie    Finance  70000 7000
4        104   Diana  Marketing  55000 5500
```

The employees tax column calculates 10% of each salary and adds it to the data. The subset() function filters out rows where the Department is "IT." The result shows the updated employees data frame with the Tax column and a version excluding IT employees.

3. Write an R function that takes a column name as input and returns all unique values of that column from a data frame. Demonstrate this using the employees data frame.

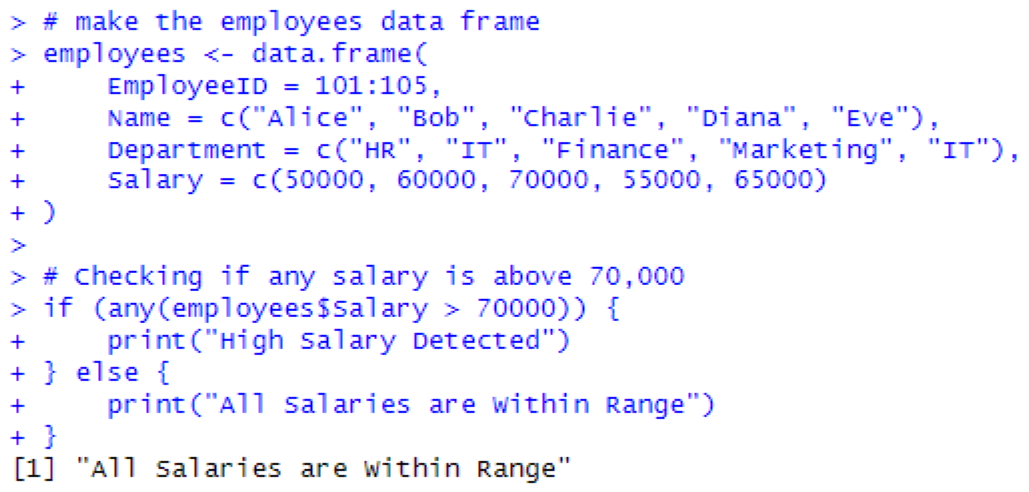


Data	
employees	5 obs. of 5 variables
non_IT_employees	3 obs. of 5 variables
Values	
even_num	int [1:25] 2 4 6 8 10 12 14 16 18 20 ...
even_numbers	int [1:25] 2 4 6 8 10 12 14 16 18 20 ...
sum_even	650L
unique_departments	chr [1:4] "HR" "IT" "Finance" "Marketing"
unique_names	chr [1:5] "Alice" "Bob" "Charlie" "Diana" "Eve"
x	int [1:50] 1 2 3 4 5 6 7 8 9 10 ...
Functions	
get_unique_values	function (data_frame, column_name)

```
> # Function to get unique values of a column from a data frame
> get_unique_values <- function(data_frame, column_name) {
+     if (column_name %in% colnames(data_frame)) {
+         return(unique(data_frame[[column_name]]))
+     } else {
+         stop("Column not found in the data frame.")
+     }
+ }
> 
> # using the employees data frame
> employees <- data.frame(
+     EmployeeID = 101:105,
+     Name = c("Alice", "Bob", "Charlie", "Diana", "Eve"),
+     Department = c("HR", "IT", "Finance", "Marketing", "IT"),
+     Salary = c(50000, 60000, 70000, 55000, 65000)
+ )
> 
> employees$Tax <- employees$Salary * 0.10
> 
> # unique values from the Department column
> unique_departments <- get_unique_values(employees, "Department")
> print(unique_departments)
[1] "HR"        "IT"        "Finance"   "Marketing"
> 
> #unique values from the Name column
> unique_names <- get_unique_values(employees, "Name")
> print(unique_names)
[1] "Alice"   "Bob"     "Charlie" "Diana"   "Eve"
```

The get_unique_values() function accepts a data frame (data_frame) and a column name (column_name) as inputs. It checks if the column exists in the data frame; if it does, it returns the unique values using the `unique()` function. If not, it shows an error. In the example, the function is used to get unique values from the `Department` and `Name` columns of the `employees` data frame.

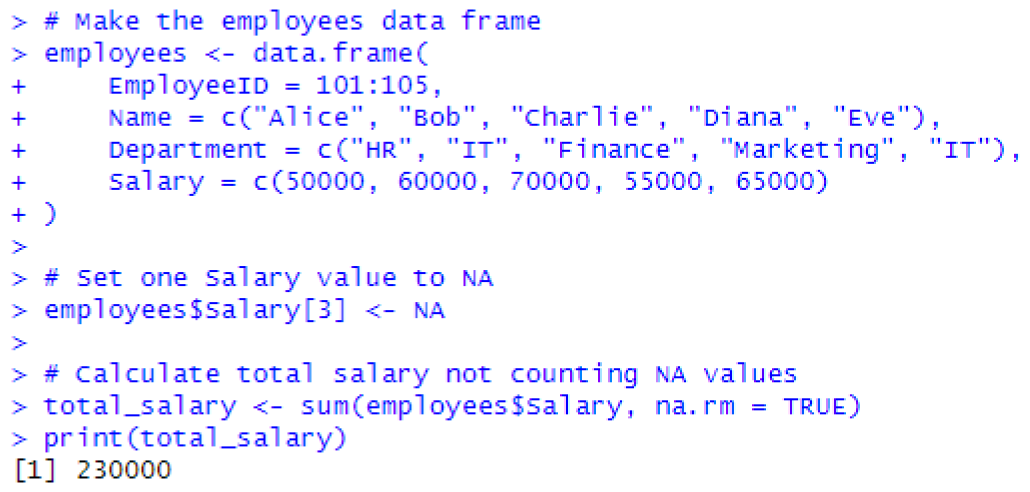
4. Write an R script to check if any Salary in the employees data frame is above 70,000. If found, print "High Salary Detected"; otherwise, print "All Salaries are Within Range".



```
> # make the employees data frame
> employees <- data.frame(
+     EmployeeID = 101:105,
+     Name = c("Alice", "Bob", "Charlie", "Diana", "Eve"),
+     Department = c("HR", "IT", "Finance", "Marketing", "IT"),
+     Salary = c(50000, 60000, 70000, 55000, 65000)
+ )
>
> # Checking if any salary is above 70,000
> if (any(employees$Salary > 70000)) {
+     print("High Salary Detected")
+ } else {
+     print("All Salaries are Within Range")
+ }
[1] "All Salaries are Within Range"
```

The any() function checks if any value in the Salary column is greater than 70,000. If any() finds a value above 70,000, it prints "High Salary Detected". Otherwise, it prints "All Salaries are Within Range".

5. Modify the employees data frame so that one Salary value is set to NA. Write code to calculate the total salary excluding the NA values and explain the output.



```
> # Make the employees data frame
> employees <- data.frame(
+     EmployeeID = 101:105,
+     Name = c("Alice", "Bob", "Charlie", "Diana", "Eve"),
+     Department = c("HR", "IT", "Finance", "Marketing", "IT"),
+     Salary = c(50000, 60000, 70000, 55000, 65000)
+ )
>
> # Set one Salary value to NA
> employees$Salary[3] <- NA
>
> # Calculate total salary not counting NA values
> total_salary <- sum(employees$Salary, na.rm = TRUE)
> print(total_salary)
[1] 230000
```

The sum() function uses na.rm = TRUE to skip NA values during the calculation. If na.rm is not used, the result will be NA because any operation with NA returns NA. The script calculates the total salary of all employees, ignoring the missing (NA) value, and prints the sum of the remaining salaries.